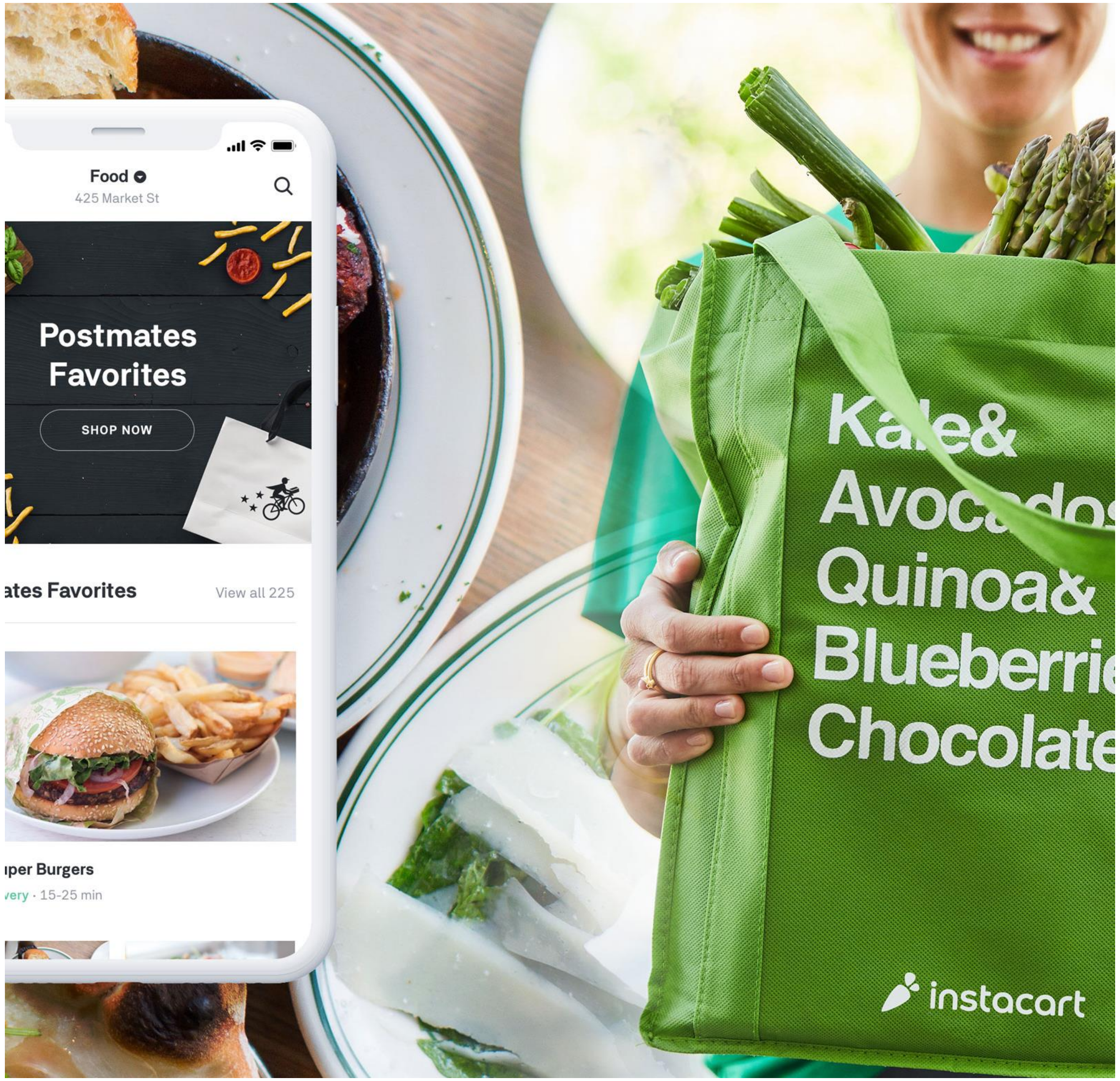
Instacart Market Basket Analysis

Project By
Valencia Dias-196004354

Introduction

- Instacart is an e-commerce website that allows users to shop for groceries from a local grocery store online, and then sends an Instacart personal shopper to pick up and deliver the orders made by users the same day.
- Instacart uses transactional data to develop models that predict which products a user will buy again, try for the first time, or add to their cart next during a session.
- These processes allow retailers to conduct analysis on purchase iterations by users but understanding the customer purchasing patterns and behaviors can become tedious and challenging.
- There are three ways that Instacart generates revenue: delivery fees, membership fees, and mark-ups on in-store prices.



Dataset Preparation

Data Source:

- **“The Instacart Online Grocery Shopping Dataset 2017” was released by Instacart as a public dataset**.
- The dataset contains over 3 million anonymized grocery orders from more than 200,000 Instacart users.
- The following analysis will make use of this datasets.
- Data source link: https://www.kaggle.com/c/instacart-market-basket-analysis/data

R Libraries Used:

- Here are the R libraries mentioned for this analysis.

Data Dictionary:

- The dataset is a relational set of files describing customers’ orders over time. They are anonymized and contains a sample of over **3 million grocery orders** from more than **200,000 Instacart users**.
- For each user, Instacart provided **between 4 and 100** of their orders, with the sequence of products purchased in each order, the **week and hour of day** the order was placed, and a **relative measure of time between orders**.

```r
library(dplyr)      # data manipulation
library(ggplot2)    # mapping variables to aesthetics,creating graphics
library(stringr)    # string manipulations
library(DT)         # R interface to Data Tables(sorting,pagination,filtering)
library(tidyr)      # tidy data
library(knitr)      # web widget
library(tidyverse)  # data manipulation
library(data.table) # fast file reading
library(caret)      # rocr analysis
library(ROCR)       # rocr analysis
library(kableExtra) # nice table HTML formatting
library(gridExtra)  # arranging ggplot in grid
library(arules)
```

Data Tables

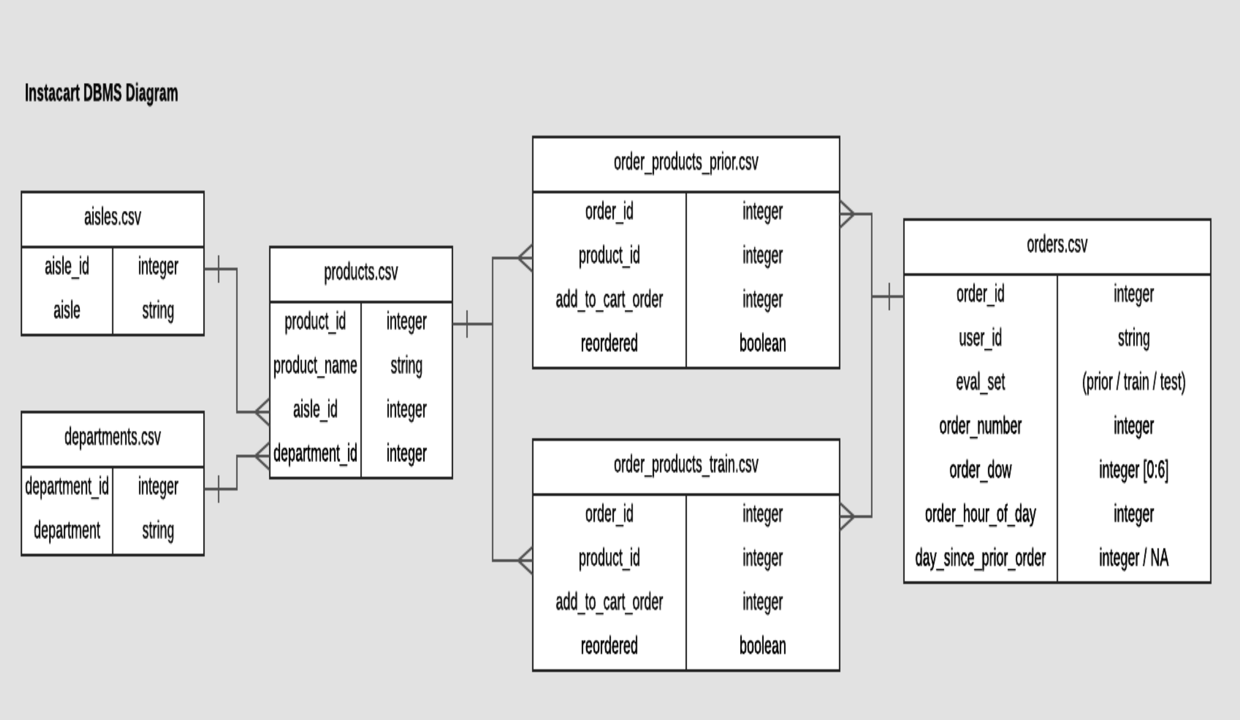
orders (3.4m rows, 206k users):
- order_id: order identifier
- user_id: customer identifier
- eval_set: which evaluation set this order belongs in (see SET described below)
- order_number: the order sequence number for this user (1 = first, n = nth)
- order_dow: the day of the week the order was placed on
- order_hour_of_day: the hour of the day the order was placed on
- days_since_prior: days since the last order, capped at 30 (with NAs for order_number = 1)

products (50k rows):
- product_id: product identifier
- product_name: name of the product
- aisle_id: foreign key
- department_id: foreign key

aisles (134 rows):
- aisle_id: aisle identifier
- aisle: the name of the aisle

departments (21 rows):
- department_id: department identifier
- department: the name of the department

order_products (30m+ rows):
- order_id: foreign key
- product_id: foreign key
- add_to_cart_order: order in which each product was added to cart
- reordered: 1 if this product has been ordered by this user in the past, 0 otherwise

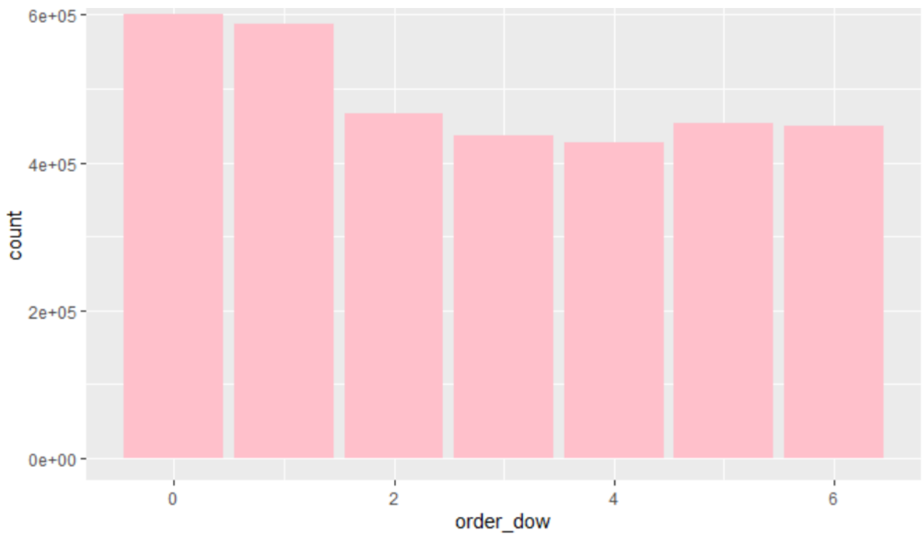


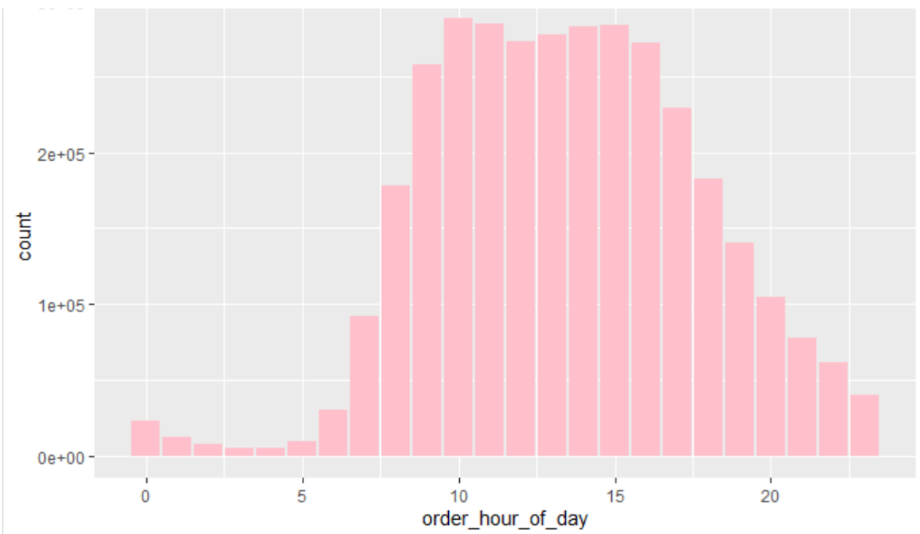
Exploratory Data Analysis

- **When do people buy?**

  -Hour of Day
  There is a clear effect of hour of day on order volume.
  Most orders are between 8.00-18.00

  -Day of Week
  There is a clear effect of day of the week.
  Most orders are on days 0 and 1.
  Unfortunately, there is no info regarding which values represent which day, but one would assume that this is the weekend



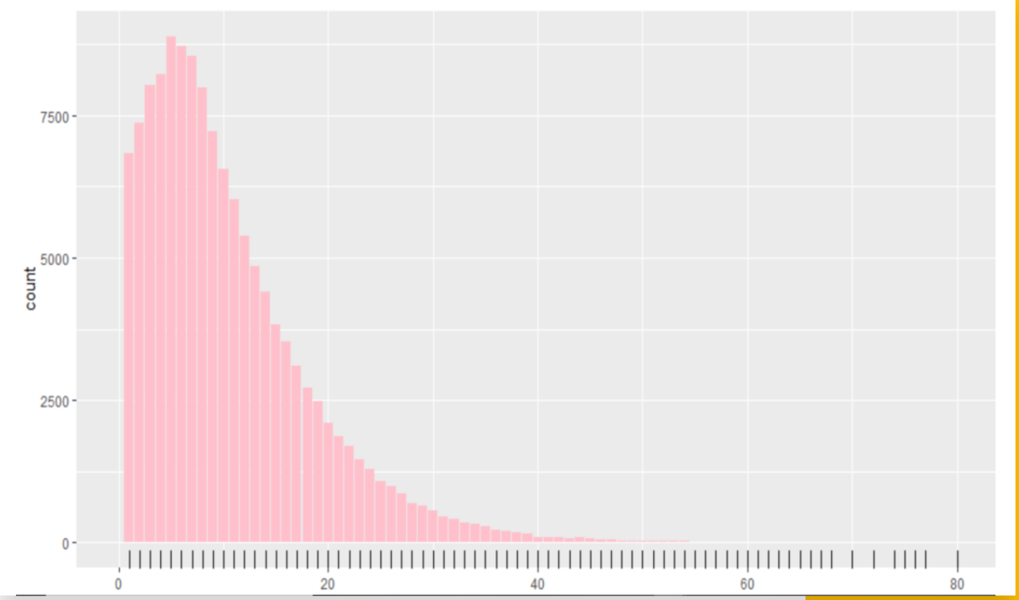


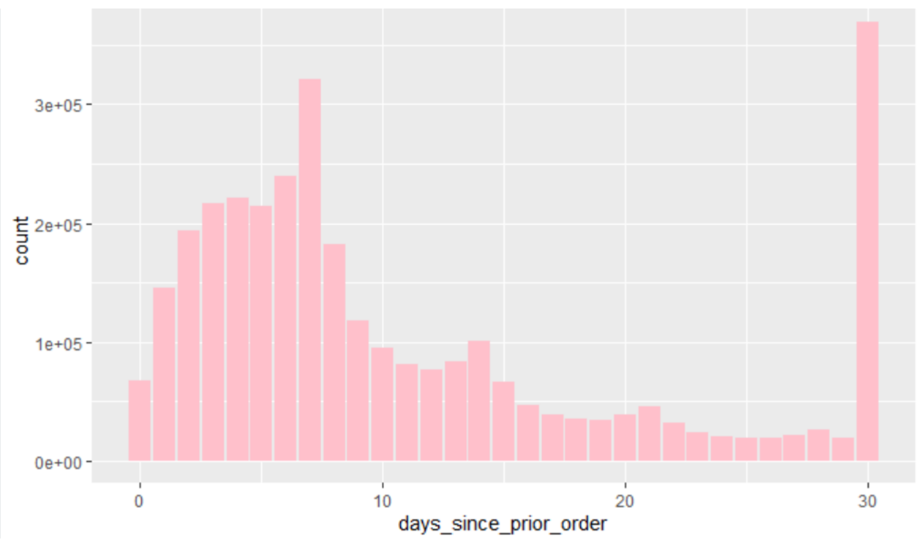
- **How many items do people buy?**

Let us have a look how many items are in the orders. We can see that people most often order around 5 items. The distributions are comparable between the train and prior order set

- **When do they order again?**

People seem to order more often after exactly 1 week.





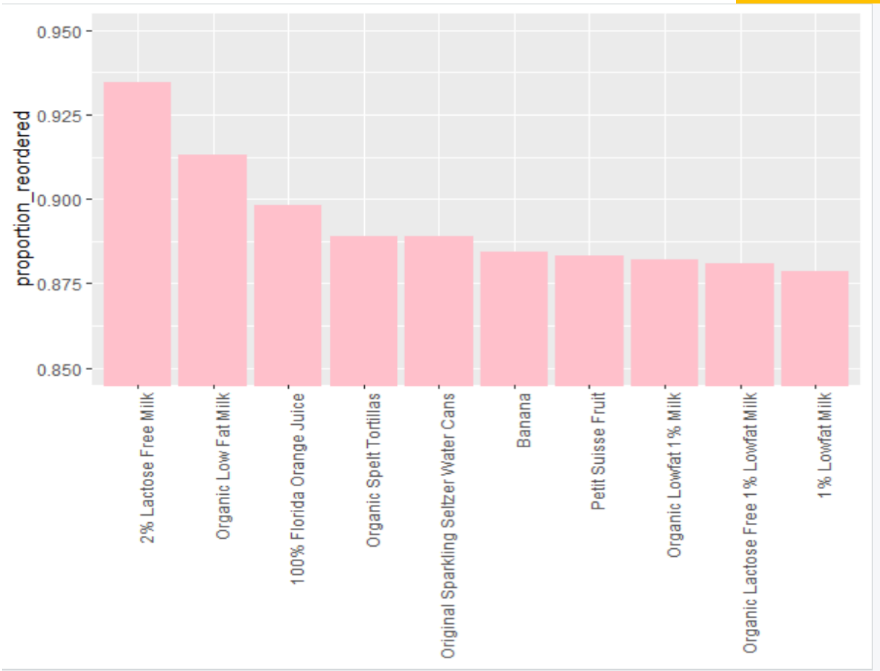
- **Most often reordered**

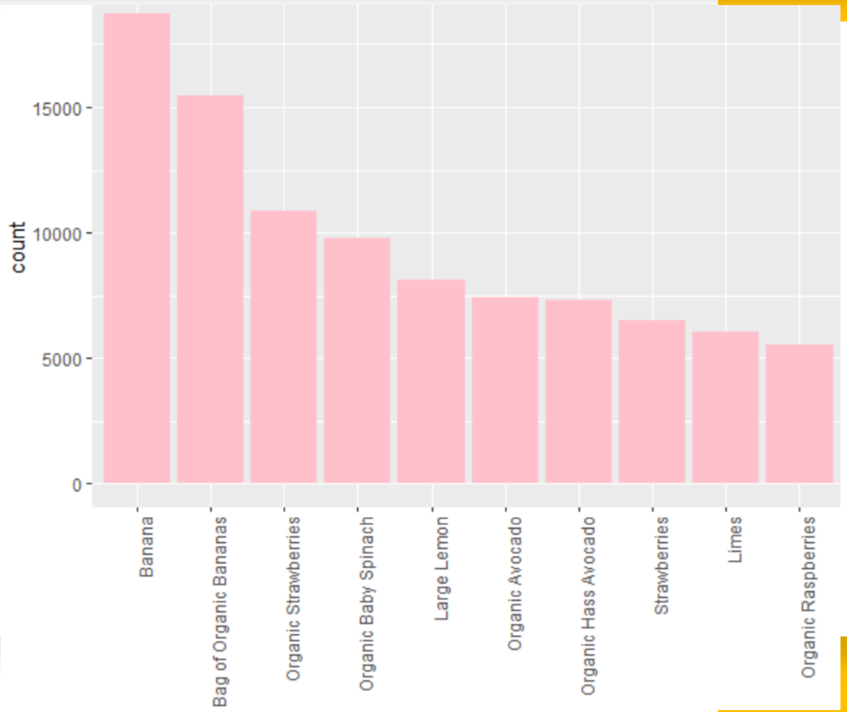
  Now here it becomes interesting.

  These 10 products have the highest probability of being reordered.

- **Bestsellers**

  Let us have a look at which products are sold most often (top10). And the clear winner is: **Bananas**



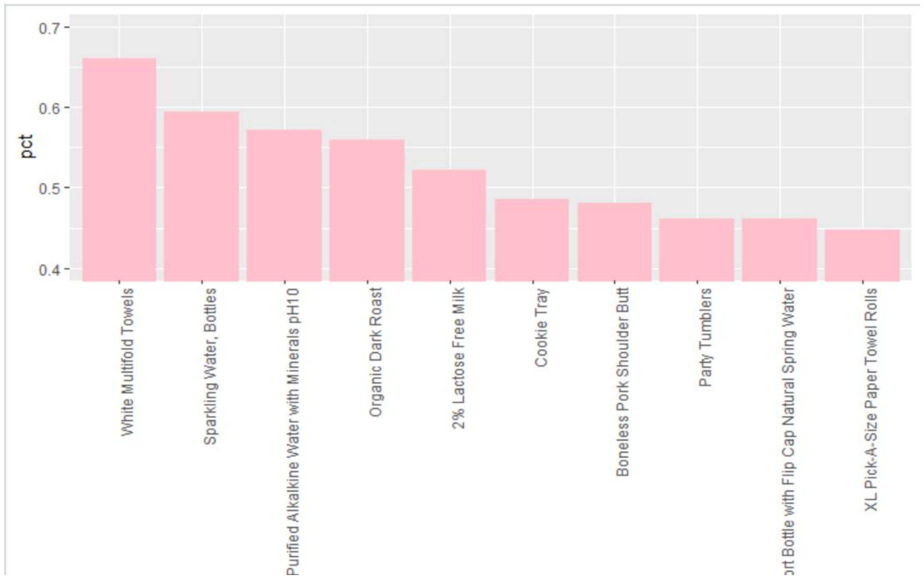


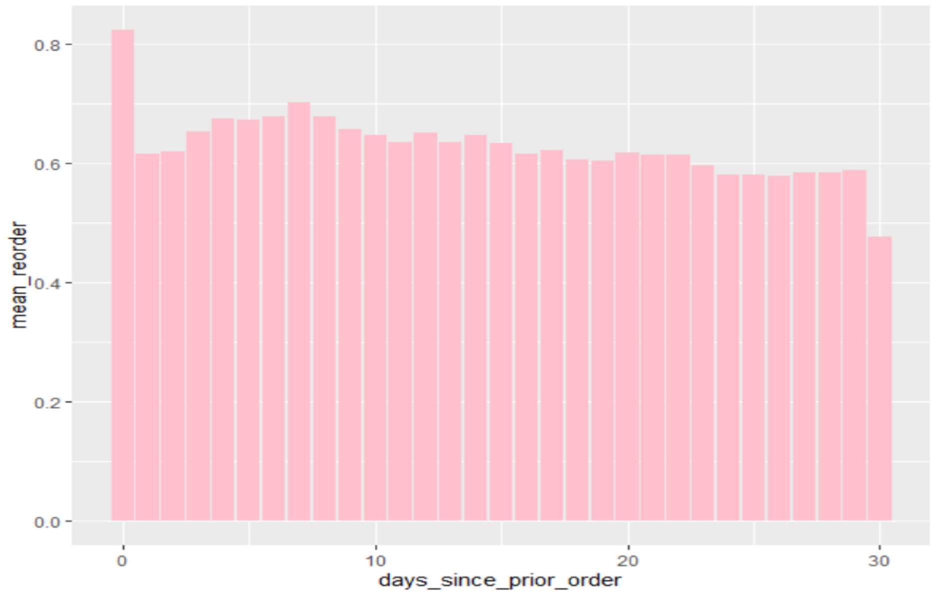
- **Which item do people put into the cart first?**

People seem to be quite certain about Multifold Towels and if they buy them, put them into their cart first in 66% of the time.

- **Association between time of last order and probability of reorder**

This is interesting: We can see that if people order again on the same day, they order the same product more often. Whereas when 30 days have passed, they tend to try out new things in their order.





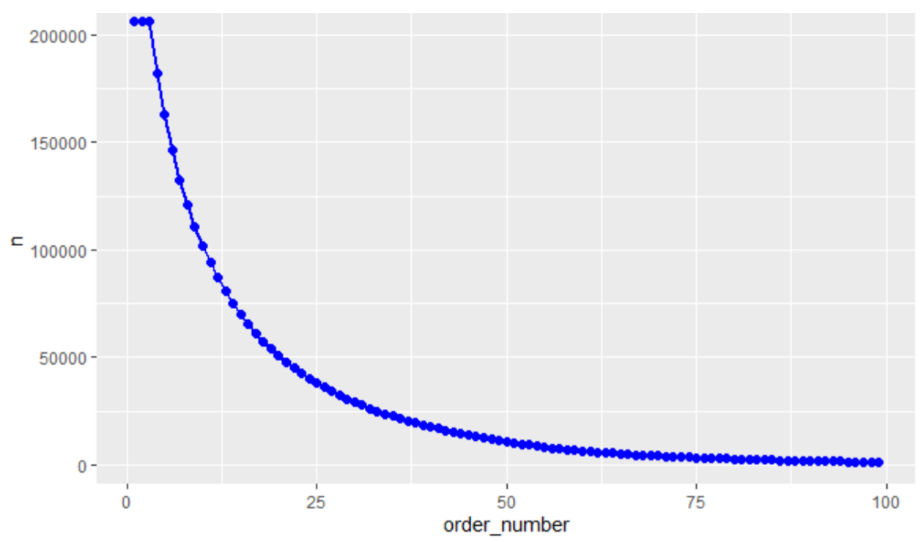
- **How many prior orders are there?**

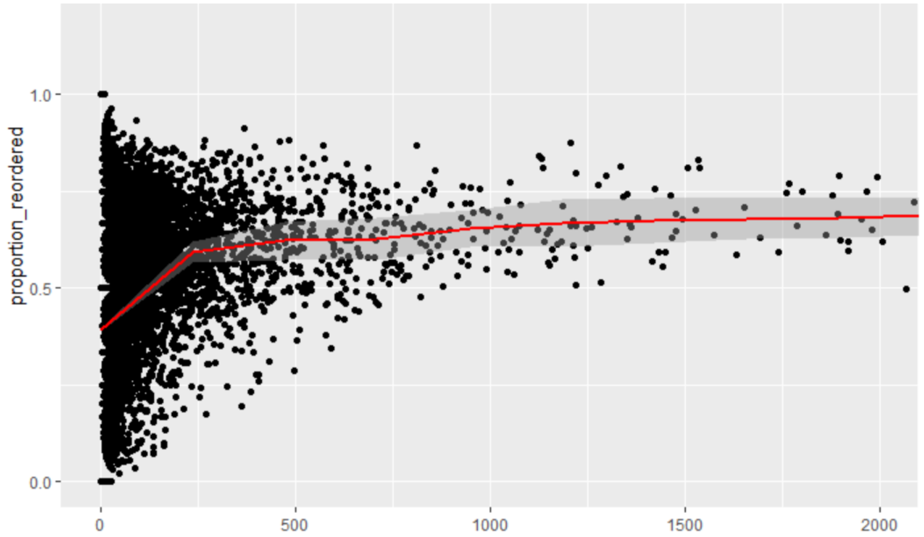
We can see that there are always at least 3 prior orders

- **Association between number of orders and probability of reordering**

Products with a high number of orders are naturally more likely to be reordered.

However, there seems to be a ceiling effect.





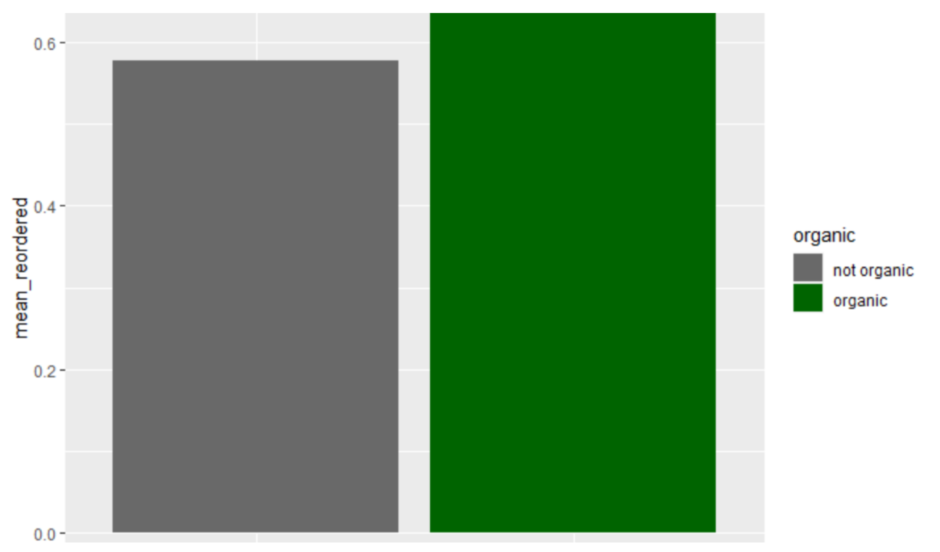
- **Reordering Organic vs Non-Organic**

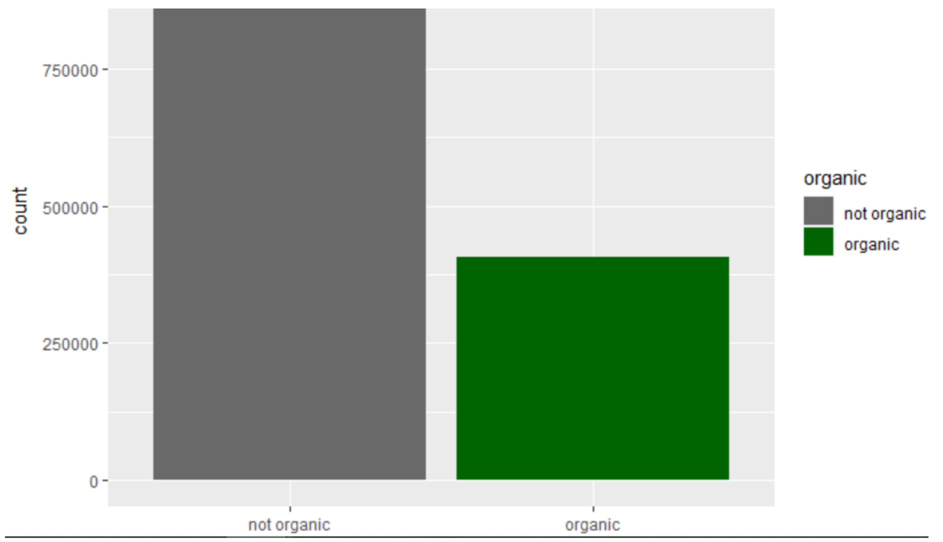
People more often reorder organic products vs non-organic products.

- **Organic Vs Non-Organic**

What is the percentage of orders that are organic vs. not organic?

People only ordered 29.29% of organic products and 70.70% of Non-Organic products.





- **Visualizing the Product Portfolio**

Here I used the treemap package to visualize the structure of InstaCart product portfolio. In total there are 21 departments containing 134 aisles

- **How are aisles organized within departments?**

- **How often are products from the department/aisle sold?**

  The size of the boxes shows the number of sales.

- **How many unique products are offered in each department/aisle?**

  The size of the boxes shows the number of products in each category.

- **Exploring Customer Habits**

Here I can see for customers who just reorder the same products again all the time. To search these, I looked at all orders (excluding the first order), where the percentage of reordered items is exactly 1 (This can easily be adapted to look at more lenient thresholds). We can see there are in fact **3,487** customers, just always reordering products.

- **The customer with the strongest habit**

The coolest customer is id #99753, having 97 orders with only reordered items. That is what I call a strong habit. She/he seems to like Organic Milk.

	user_id	n_equal	percent_equal
1	99753	97	1
2	55331	49	1
3	106510	49	1
4	111365	47	1
5	74656	46	1
6	170174	45	1
7	12025	43	1
8	164779	37	1
9	37075	34	1
10	110225	33	1

	order_id	product_id	add_to_cart_order	reordered	product_name
1	46614	27845	1	1	Organic Whole Milk
2	46614	38689	2	1	Organic Reduced Fat Milk
3	67223	27845	1	1	Organic Whole Milk
4	67223	38689	2	1	Organic Reduced Fat Milk
5	214506	27845	1	1	Organic Whole Milk
6	214506	38689	2	1	Organic Reduced Fat Milk
7	240832	27845	1	1	Organic Whole Milk
8	240832	38689	2	1	Organic Reduced Fat Milk
9	260804	27845	1	1	Organic Whole Milk
10	260804	38689	2	1	Organic Reduced Fat Milk

Predictive Analytics

Type of Prediction

- The objective is to predict what product will the customer purchase in the next basket. It requires probability estimation of each product that has been purchased before, that to be purchased before. **This is a classification problem**, as well as **a regression of probability** of repurchases.
- For this analysis, I have used two Naive models (handcrafted baseline) and one Machine Learning Logistic regression will be used for Machine Learning approach for its speed and simplicity; to demonstrate the feasibility to producing a better outcome then baseline.

Train/Test Dataset Splitting

- Instacart did not provide us **test order detail**, therefore we shall use the **train** users for both training and testing.
- We achieve this by splitting the **train** users and its related orders and products into train dataset and train dataset, at 70%/30% split (by number of users). That means our train/test dataset will contain approximately 91846 / 39,363 users.

```r
# update this variable for changing split ratio
train_proportion = 0.7
# build list of all users ID
tmp = orders %>% filter(eval_set=='train') %>% distinct(user_id)
# 70/30 split
set.seed(12345)
train.rows = sample( 1:nrow(tmp), train_proportion * nrow(tmp) )
train.users = tmp[train.rows,]  # select training rows, list of train users
test.users  = tmp[-train.rows,] # select testing rows, list of test users
cat("Total Rows in Training Users: ", length(train.users),
    "\nTotal Rows in Testing Users: ", length(test.users),
    "\nTrain/Test Split % : ", 100*length(train.users)/(length(test.users)+length(train.users)),
    " / ", 100*length(test.users)/(length(test.users)+length(train.users)))
```

```
Total Rows in Training Users:  91846
Total Rows in Testing Users:  39363
Train/Test Split % :  69.99977  /  30.00023
```

Model Evaluation & Optimization

- Instacart has close to 50k products in their catalogue. As the maximum number of items ordered by a user is just a fraction of the 50k available product. This means by simply predicting nothing is purchased in the next basket, we would yield **close to 100% accuracy**.

- Due to the **highly imbalance** dataset, Instacart require **F1 Score** as the competition scoring, instead of accuracy. To evaluate the performance of the model, we had created a custom function to build a **confusion matrix and derive other binary classification metrics**.

Model 1: Naive Prediction

- With intension to make this a baseline model, simply predict the basket based on user last order.

- The result shows only **0.3460833 F1 Score**.

```
##Model 1 : Naive Prediction
m1.train.data = users_orders_products_ %>%
  filter(user_id %in% train.users) %>%
  group_by(user_id) %>%
  top_n(n=1, wt=order_number)  %>% #last order has the higher order_number
  select(user_id, product_id) %>%
  mutate (predicted=1)  %>%        #predict based on last ordered, therefore 1
  full_join(train.construct) %>%   #join with train construct for items not
                                   #predicted but in final order
  select(user_id, product_id, actual, predicted) %>%
  replace_na(list(predicted = 0))
head(m1.train.data,25)
```

```
> m1.eval = binclass_eval(m1.train.data$actual, m1.train.data$predicted)
> m1.eval$cm
      Predicted
Actual       0       1
     0 4662319  687076
     1  314988  265933
> cat("Accuracy:  ", m1.eval$accuracy,
+     "\nPrecision: ", m1.eval$precision,
+     "\nRecall:    ", m1.eval$recall,
+     "\nFScore:    ", m1.eval$fscore)
Accuracy:   0.8310269
Precision:  0.2790456
Recall:     0.4577783
FScore:     0.3467342
```

Model 2: Smarter Naive Prediction (Baseline)

- In this model, we predict products in the basket by estimating their frequency of repurchased. This way we get a ratio to indicate probability of re-purchases. We use ROCR package to estimate the best cutoff point (at which above this cutoff we shall predict for re-order) that give us the **optimum F1 score**.
- We are getting slightly **better F1 Score (0.3753544)** compare to previous naive model. We shall use this as the **BASELINE**.

```
## Build Model
m2.train.data = users_orders_products_ %>%
  filter(user_id %in% train.users) %>%
  group_by(user_id) %>%
  mutate(total_orders = max(order_number)) %>%  # total number of orders made previously
  ungroup %>%
  select(user_id, order_id, product_id, total_orders) %>%
  group_by(user_id, product_id) %>%
  summarize(predicted=n()/max(total_orders)) %>%
  select(user_id, product_id, predicted) %>%
  full_join(train.construct) %>%  # join with train construct for items not predicted but in final
  select(user_id, product_id, actual, predicted) %>%
  replace_na(list(predicted = 0))
head(m2.train.data,20)
```

```
> ### Threshold Optimization
> m2.rocr = optimize_cutoff(actual = m2.train.data$actual, probability = m2.train.data$predicted)
> kable(m2.rocr$best) %>% kable_styling(bootstrap_options = c("striped"))
> m2.eval = binclass_eval(m2.train.data$actual, m2.train.data$predicted>0.3367347)
> m2.eval$cm
      Predicted
Actual       0       1
     0 5012298  337097
     1  368728  212193
> cat("Accuracy:  ", m2.eval$accuracy,
+     "\nPrecision: ", m2.eval$precision,
+     "\nRecall:    ", m2.eval$recall,
+     "\nFScore:    ", m2.eval$fscore)
Accuracy:   0.8809802
Precision:  0.3863041
Recall:     0.36527
FScore:     0.3754927
```

Machine Learning Framing

We construct all the products that users had purchased in the last 3 orders, then use machine learning classification to predict will each of the product be purchased again.

We shall use **decision tree and logistic regression** for this prediction.

Model 3: Logistic Regression

- Training of the model
- Prediction
- Building the confusion matrix for Train and Test data

```
#Model Training
m3.fit = glm(actual ~ ., family = binomial, data = m3.train.data)

#Prediction
m3.predict = predict(m3.fit, type = 'response', newdata = m3.train.data)

#Confusion Matrix
m3.eval = binclass_eval(m3.train.data$actual, m3.predict>0.2233115)
m3.eval$cm
```

	Predicted	
Actual	0	1
0	9191429	2312354
1	1022350	1948260

	Predicted	
Actual	0	1
0	3921528	978677
1	435702	831691

Model Evaluation

- Logistic regression produces F1 Score of 0.5388937 with training data, a much better compared to Model 1 and Model 2.
- We shall proceed to test the model on unknown data, the test data.
- We achieved F1 Score of **0.5405588**, slightly higher than training data.

```
Accuracy:   0.7696136
Precision:  0.4572721
Recall:     0.6558451
FScore:     0.5388465
```

```
Accuracy:   0.7706759
Precision:  0.4594044
Recall:     0.6562219
FScore:     0.540452
```

- **ROC**

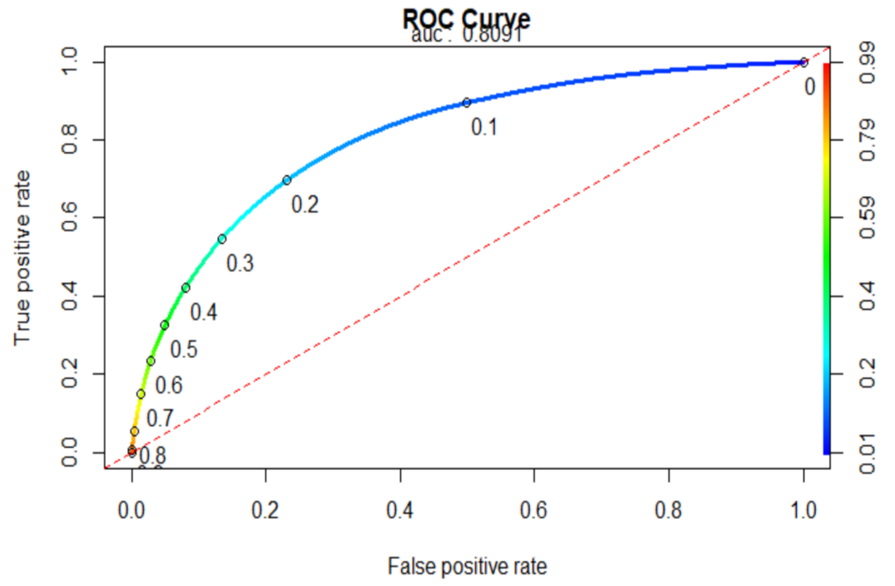
```
#ROC
rocr.auc
```

```
[1] 0.8090974
```

```
#ROC
rocr.auc

plot(rocr.perf,
     lwd = 3, colorize = TRUE,
     print.cutoffs.at = seq(0, 1, by = 0.1),
     text.adj = c(-0.2, 1.7),
     main = 'ROC Curve')
mtext(paste('auc : ', round(rocr.auc, 5)))

abline(0, 1, col = "red", lty = 2)
```



Conclusion

- Doing this analysis allowed me to understand the fine-grained details about the customer shopping behavior on the Instacart platform.
- Knowing which items are most frequently purchased is the first step for Instacart to optimize its software product and recommend items for customers while they shop.

Citations

- https://www.kaggle.com/c/instacart-market-basket-analysis/data
- http://blog.kaggle.com/2017/09/21/instacart-market-basket-analysis-winners-interview-2nd-place-kazuki-onodera/
- https://meiyipan.com/2017/09/16/instacart/
- http://cs229.stanford.edu/proj2017/final-reports/5240337.pdf


Thank you